IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	**10/035,905**
Appellant	:	**MONTGOMERY, Michael**
Filing Date	:	**12/24/2001**
Confirmation No	:	**2189**
Art Unit	:	**2876**
Examiner	:	**FRECH, Karl**
Docket No.	:	**40.0049**
Customer No.	:	**41754**

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF

1. Real Party in Interest

The real party in interest in this appeal is Gemalto N.V., a corporation of the Netherlands. The application is formally assigned to Axalto, Inc. Axalto, Inc. is the U.S. subsidiary of Gemalto.

2. Related Appeals and Interferences

There are no related appeals and interferences.

3. Status of Claims

Claims 1-27 are pending in the application. Claims 1-27 were rejected in the Office Action of 12/29/2005.

4. Status of Amendments

No amendments have been made since the Office Action of 12/29/2005.

5. Summary of Invention

The present invention seeks to optimize the processing of transactions within a data processing system by minimizing the number of commits required for transactions that have successfully completed. In systems (such as smart cards) utilizing persistent, non-volatile memory such as EEPROM, the invention may (as a result of minimizing the number of commits) increase the operational life of such persistent, non-volatile memory. The invention utilizes a change in the observable state of the system performing the transactions in order to consolidate two or more transactions that have completed successfully and commit the resulting data of the consolidated transactions together rather than committing the resulting data of each transaction separately. A change of observable state is a change in the data processing system that could be observed by an outside observer, e.g., a user or another computer program that operates on data processed by the data processing system.

A transactional operation is a set of instructions that must (as a set) succeed completely. In principle, if a failure (computational or otherwise) occurs before the set of instructions has succeeded completely, the system's values must be restored to the state

they were in before the transaction operation was attempted. However, in Appellants' invention, transactions are grouped together based on observable state of each of the transactions in a set. Appellants made the observation that if there were no a change in observable state after a given transaction, there would be no need to be able to rollback to that particular transaction. Rather a rollback could be to an earlier transaction after which there indeed was a change in observable state.

In Claim 1, the invention is recited as "A method for processing a plurality of transactions within computer code being executed by a data processing system comprising:

a) examining the computer code being executed for a transaction after the execution of which a change in observable state could occur;

b) storing data for the executed computer code that are part of a sequence having a plurality of transactions between a location before the sequence and the transaction after the execution of which a change in observable state could occur, wherein the data is stored within the computer code; and

c) responsive to detecting a change in observable state, committing a portion of the stored data to non-volatile memory."

In other words, a process implementing the claimed invention commences by determining a transaction after the execution of which a change in observable state could occur. Up to that point, the process stores within the computer code any data for the sequence of transactions that end with the transaction identified as a candidate for having a change in observable state. Then, if a change of observable state does occur at that point, the process commits a portion of the stored data to non-volatile memory.

6. Grounds for Rejection to be Reviewed on Appeal

1. 35 USC 112

Claims 1-27 were rejected under 35 USC 112, second paragraph as indefinite for failing to particularly point out and distinctly claim the subject matter which appellant regards as the invention.

2. 35 USC 102(b)

Claims 1-4, 11-14, 21-24 were rejected under 35 U.S.C. 102(b) as being anticipated by Cable (US Patent No. 5,999,728, hereinafter Cable).

3. 35 USC 103(a)

Claims 5-10, 15-20, 25-27 were rejected under 35 U.S.C. 103(a) as being unpatentable over Cable in view of well known prior art.

7. Argument

1. 35 USC 112, second paragraph

Claims 1-27 stand rejected under 35 USC 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which appellant regards as the invention. Appellants respectfully request reversal of this rejection and allowance of the Claims.

The Examiner asserted that the recitation of the term "could occur" renders the claims indefinite "as the term 'could occur' is conditional and does not define the claimed invention" (Office Action, Page 2, Numbered Paragraph 3). The Examiner continues with "[i]n the instant case, the examiner asserts that 'could occur' also suggests to one of ordinary skill 'could not occur' is also contemplated." Appellant posit that the use of "could occur" does not render the claim indefinite.

Claim 1 recites the step of "a) examining the computer code being executed for a transaction after the execution of which a change in observable state could occur". This step is perfectly definite. It recites a condition that defines a transaction for which a process implements the claims the method should examine the computer code.

Consider an analogy. Suppose that an attorney has a fondness for photographing cherry blossoms and occasionally travel to Washington D.C. on patent office business. The attorney may then wish to mark his calendar with dates when the cherry trees could blossom so that he may ensure that he packs a camera on any trips to the patent office on such dates. A method step for this would state "mark calendar on days on which cherry blossoms could occur." All persons know that would exclude summer, fall and winter days. The National Park Service defines the blooming period to be March 25 to April 8th, http://www.nps.gov/nacc/cherry/updated.htm. Suppose that there is a non-zero chance of blooms one week in either direction. Then, March 17th through April 15th, having the possibility of cherry blooms, must be marked. True, some of those days may not have cherry trees in bloom. Yet, a person with ordinary skill would know that those days still need to be marked in the calendar.

Similarly here, not all transactions result in a change in observable state. Others have the possibility of resulting in a change in observable state. It is crystal clear which instructions would be examined for; namely, those "after the execution of which a change in observable state could occur". Other transactions, i.e., those where a change of observable state could not occur, are not sought.

Next, Claim 1 recites "storing data for the executed computer code that are part of a sequence having a plurality of transactions between a location before the sequence and the transaction after the execution of which a change in observable state could occur, wherein the data is stored within the computer code." In the preceding step, the computer code was examined for a transaction after the execution of which a change in observable state could occur. Now, having identified such a candidate for change in observable state, data is stored within the computer during the intervening sequence of transactions. Thus, having identified a precise transaction in step (a), the method proceeds to that precise transaction.

Consider the analogy again. Suppose the attorney with a fondness for cherry-blossom photography wishes to postpone a trip to the first day when he will encounter cherry trees in bloom. The attorney may then follow the instruction "postpone trip until first date when cherry blossoms could occur." If we are now in July, we know that the next half-year would be out of the question. However, with the assumption made above, come March 17th, the cherry blossoms could occur and the attorney would travel to arrive on that date. There would be no doubt as to the required travel date. The method for establishing the travel date is one hundred percent definite, even if the cherry-blossoms may or may not be out yet.

Similarly here, while it should not be possible to assess whether a change of observable state will occur on the identified transaction, the method is totally definite as to for which transactions data should be stored in the computer code without making a commit to non-volatile memory.

Thus, the use of the term "could occur" imparts no indefiniteness on the claim. Accordingly, the rejection under 35 USC 112, second paragraph should be reversed and the claim allowed.

2. 35 USC 102(b)

Claim 1-4, 11-14, 21-24 stand rejected under 35 USC 102(b). Appellants respectfully request reversal of the rejection and allowance of the claim.

Preliminarily, Appellants wish to point out that the Examiner has deemed Appellants' arguments based on the limitations "could occur" as not persuasive because "[as] 'could occur' also implies 'could not occur' the examiner must interpret both situations." Office Action, Page 4, Numbered Paragraph 9. As discussed hereinabove in conjunction with the argument in regard to the 35 USC 112, second paragraph rejection, that view is not valid. The limitation "could occur" precisely define the transactions that need to be examined for and that limit the sequence of transactions that do not require a commit to non-volatile memory.

That said, Appellants respectfully request that the Board consider the arguments presented in the arguments filed 10/19/2005. Those arguments are re-presented here.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. The

identical invention must be shown in as much detail as is contained in the claim." MPEP 2131. That standard cannot be met using the Cable reference.

Cable addresses an entirely different problem from the problem solved by the appellants. It is therefore not surprising that Cable does not teach or suggest appellants' novel and non-obvious invention as recited in the claims.

A summary of Cable may aid the analysis under 35 USC 102(b). Cable provides a system for porting a toolkit of a graphical user interface from one window-based platform to another window-based platform. Cable's method addresses the problem of portability that arises "because the interactions between objects are defined by a model-view-controller developed as a platform specific implementation for use with a specific window-based environment, the toolkit cannot be readily ported from one window-based platform to another." Cable, Col. 2, Lines 36-40. Furthermore, Cable states "[t]he amount of computer code which must be revised to use a toolkit defined for one window-based platform with another window-based platform is therefore excessive and impractical." Cable, Col. 2, lines 52-55. Therefore, Cable states, "it would be desirable to abstract the events received from a window-based system so that the primary portions of code used to represent the toolkit of an object oriented graphical user interface can remain unchanged, and therefore be easily ported to any of a variety of window based platforms." Cable, Col.2, line 65-Col. 3, line 3.

As part of the solution for the portability of a windows-based graphic user interface toolkit, Cable translates native notifications implemented with respect to a first window-based platform into abstracted notifications for use in defining the graphical user

interface. "As a result, each object of the graphical user interface toolkit can be registered with an abstracted notification received from any of plural window-based platforms. Cable, Col. 4, lines 30-40. As Cable deals with graphical user interfaces, naturally, "the notifications can correspond an event such as a key stroke, activation of a push button, an iconified window, and so forth". Cable, Col. 4, lines 41-43. "The functional signature used to represent an abstract notification constitutes a behavioral specification ... to which model, view, and controller object implementations can be made to conform". Col. 4, lines 48-51.

Appeallants address an entirely different problem. "In one embodiment of the present invention, changes in the observable state of the system may be utilized to consolidate transactions and commit the applicable data of the consolidated transactions together because the client will not rely upon such data until a change in the observable state occurs." Specification, Page 6, Lines 1-4. In other words, Appellants present a solution in which the number of writes to a non-volatile memory is minimized by avoiding writes to that memory at the conclusion of transactions other than when a change in observable state could have occurred. It should be noted that in the context of the present invention a transaction is a sequence of operations that must either complete successfully or that may be rolled back to the status quo ante. Similarly, a change in observable state, in the context of the invention, occurs when the current state of the system would be available to those accessing the system from the outside world.

As Appellants observe in the Background:

"Many data processing systems (also referred to herein as "systems") require transactional operations to function successfully. A transactional operation is a set of instructions that must (as a set) succeed completely. If a failure (computational or otherwise) occurs before the set of instructions has succeeded completely, the system's values must be restored to the state they were in before the transaction was attempted." Specification, Page 1, lines 12-17.

To ensure the "transactional" characteristic of a series of operations, the beginning of the series of operations is marked, the individual operations performed, and at the end the results from the operations are committed (i.e., the system state is updated). If the transactions fail, the state of the system can be rolled back to the state at the beginning of the sequence of operations that make up a transaction.

"Generally speaking, the invention contemplates optimizing the processing of transactions within a data processing system by minimizing the number of commits required for transactions that have successfully completed. In systems (such as smart cards) utilizing persistent, non-volatile memory such as EEPROM, the invention may (as a result of minimizing the number of commits) increase the operational life of such persistent, non-volatile memory. The invention utilizes a change in the observable state of the system performing the transactions in order to consolidate two or more transactions that have completed successfully and commit the resulting data of the consolidated transactions together rather than committing the resulting data of each transaction separately." Specification, Page 5, lines 10-18.

"A person skilled in the art will appreciate that changes in observable state generally include any means by which the state of the system is made available to the outside world in such a way that those accessing the services of the system could

reasonably become aware of the state." Specification, page 5, lines 20-23. "From the viewpoint of the client, the observable state of the system may be defined to generally include only that part of the externally visible state of the system as perceived by the client. " Specification, page 5, lines 31-34.

From the foregoing, it should be apparent that Cable and Appellants address entirely different problems. Therefore, it should not come as a surprise, that Cable does not teach or suggest the invention claimed by Appellants. Appellants claim, for example, "examining the computer code being executed for a transaction after the execution of which a change in observable state could occur". Cable does not teach or suggest such a limitation.

For that proposition, the Examiner states in the office action that "Cable discloses in column 4 lines 43+ that state changes in a window's based platform are encapsulated in abstract notifications and, in column 6 lines 59+ that the abstract notifications can be stored in a readable memory medium." Office Action, Page 2, numbered paragraph 5. It should be noted, however, that Cable's state changes have no relationship to transactions, wherein a transaction is a sequence of operations that must either complete correctly or that may be rolled back to a prior state. Thus, a mention in Cable of state changes is not equivalent to Appellants' "examining the computer code being executed for a transaction after the execution of which a change in observable state could occur" (Claim 1) and certainly does not set forth this element in as much detail as contained in the claim. Furthermore, the cited passage of Cable does not suggest "examining the computer code

being executed for a transaction after the execution of which a change in observable state could occur".

Appellants further claim (in claim 1) that "storing data for ... a sequence ... of transactions between a [first] location ... and the transaction after the execution of which a change in observable state would occur, wherein the data is stored within the computer code" and "responsive to detecting a change in observable state, committing a portion of the stored data to non-volatile memory." In other words, the data associated with a sequence of transactions – again please consider that a transaction has the specific meaning of a series of operations that either must complete correctly or be rolled back – is stored within the computer code until an observable state may have occurred. Intermediate data is not committed to the non-volatile memory.

The Examiner has made the statement that "That is, Cable discloses finding state changes in computer code and storing data elative to the state change in the computer code" (Office Action, Page 2, Numbered Paragraph 5). While the Appellants make no claim to storing data per se, it should be noted that the claim limitation sets forth that data is stored in non-volatile memory in response to a change in observable state and not for transactions preceding that event. A general teaching in a reference that some data may be stored in a computer readable storage medium is not specific enough to constitute a teaching of the specific limitations recited in Appellants novel and non-obvious claim.

For these reasons, Cable does not teach or suggest each and every element set forth in the claim; much less "in as much detail as is contained in the claim".

Accordingly, Claim 1 is not anticipated by Cable and should be allowed. Claims 11 and 21 recite analogous limitations and are therefore patentable over Cable for the same reasons given in support of Claim 1.

Claims 3-4, 12-14, and 22-24 depend from Claims 1, 11, and 21, respectively, incorporate the limitations of their respective base claims, set forth further unique and non-obvious combinations, and are, therefore, patentable for the reasons given in support of Claims 1, 11, and 21 and by virtue of such further combinations.

3. 35 USC 103(a)

Claims 5-10, 15-20, and 25-27 stand rejected under 35 USC 103(a) as unpatentable over Cable. Appellants respectfully request reversal of the rejection and allowance of the claims.

The Examiner has failed to establish a prima facie case of obviousness. "To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations." MPEP 2143. The Examiner has failed to meet this burden.

As argued hereinabove with respect to the rejection under 35 USC 102, Cable does not teach or suggest the invention as set forth in Claims 1, 11, and 21.

Therefore, at least the third requirement for a *prima facie* case of obviousness has not been met with respect to the underlying base claims. Because Claims 5-10, 15-20, and 25-27 depend from Claims 1, 11, and 21, respectively, incorporate the limitations of their respective base claims, set forth further unique and non-obvious combinations, *prima facie* case of obviousness has not been set forth with respect to these claims for the reasons given in support of Claims 1, 11, and 21 and by virtue of such further combinations.

"If examination at the initial stage does not produce a prima facie case of unpatentability, then without more the appellant is entitled to grant of the patent." In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992), *quoted in* In re Lowry, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994). Thus, for the reasons given above, Appellants respectfully request reversal of the rejection of Claims 1,16, 29, 59, 62, and 77 and their early allowance.

Accordingly, Appellants respectfully requests reversal of the rejections and allowance of the claims.

Conclusion of Argument

Appellants have argued hereinabove that the rejections under 35 USC 112, second paragraph, 35 USC 102(b), and 35 USC 103(a) are all improper and that the appliation

meets the requirements under 35 USC 112, second paragraph, and is patentable over the prior art. Accordinly, Appellants respectfully request reversal of the rejection of Claims 1-27 and their early allowance.

Respectfully Submitted,

/Pehr Jansson/

Pehr Jansson
Reg. No. 35,759

Date: July 28, 2006

Pehr Jansson
Anderson & Jansson, L.L.P.
9501 N. Capital of TX Hwy.
Austin, TX 78759
512-372-8440 x200
512-233-3447 (fax)
pehr@anjanlaw.com

8. Claims appendix

1. A method for processing a plurality of transactions within computer code being executed by a data processing system comprising:

a) examining the computer code being executed for a transaction after the execution of which a change in observable state could occur;

b) storing data for the executed computer code that are part of a sequence having a plurality of transactions between a location before the sequence and the transaction after the execution of which a change in observable state could occur, wherein the data is stored within the computer code; and

c) responsive to detecting a change in observable state, committing a portion of the stored data to non-volatile memory.

2. The method of claim 1 wherein the portion of the stored data committed includes stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

3. The method of claim 1 further comprising:

d) responsive to detecting that execution of a transaction causes a system limitation in the data processing system to occur, committing a portion of the stored data.

4. The method of claim 3 wherein the portion of the stored data committed includes stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

5. The method of claim 1 further comprising:

d) maintaining a pointer to the logical beginning of the stored data and maintaining a pointer to the end of the stored data for the last transaction within the computer code to be fully executed.

6. The method of claim 1 wherein the stored data are stored in a transaction buffer.

7. The method of claim 6 wherein the transaction buffer comprises random access memory.

8. The method of claim 1 wherein new value logging is utilized.

9. The method of claim 1 wherein old value logging is utilized.

10. The method of claim 1 wherein the data processing system comprises a smart card.

11. A data processing system configured to execute computer code having a plurality of transactions within the computer code comprising:

a memory;

a processor connected to the memory; and

having logic to cause the processor to process the plurality of transactions by a) examining the computer code being executed for a transaction after the execution of which a change in observable state could occur; b) storing data for the executed computer code that are part of a sequence having a plurality of transactions between a location before the sequence and the transaction after the execution of which a change in observable state could occur, wherein the data is stored within the computer code; and c) responsive to detecting a change in observable state, committing a portion of the stored data.

12. The data processing system of claim 11 further having logic to cause the portion of the stored data committed to include stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

13. The data processing system of claim 11 further having logic to cause the processor to process the plurality of transactions by d) responsive to detecting that execution of a transaction causes a system limitation in the data processing system to occur, committing a portion of the stored data.

14. The data processing system of claim 13 further having logic to cause the portion of the stored data committed to include stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

15. The data processing system of claim 11 further having logic for maintaining a pointer to the logical beginning of the stored data and maintaining a pointer to the end of the stored data for the last transaction within the computer code to be fully executed.

16. The data processing system of claim 11 further comprising a transaction buffer for storing the data.

17. The data processing system of claim 16 wherein the transaction buffer comprises random access memory.

18. The data processing system of claim 11 further having logic for utilizing new value logging.

19. The data processing system of claim 11 further having logic for utilizing old value logging.

20. The data processing system of claim 11 wherein the data processing system comprises a smart card.

21. A computer-readable medium tangibly having a program of machine-readable instructions for causing a processor to perform a method for processing a plurality of transactions within computer code being executed by a data processing system, the method comprising:

a) examining the computer code being executed for a transaction after the execution of which a change in observable state could occur;

b) storing data for the executed computer code that are part of ~~the~~ a sequence having a plurality of transactions between a location before the sequence and the transaction after the execution of which a change in observable state could occur, wherein the data is stored within the computer code; and

c) responsive to detecting a change in observable state, committing a portion of the stored data.

22. The computer-readable medium of claim 21 further having instructions for causing the portion of the stored data committed to include stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

23. The computer-readable medium of claim 21 further having instructions for causing a processor to perform a method for processing a plurality of transactions within computer code being executed by a data processing system, the method comprising:

d) responsive to detecting that execution of a transaction causes a system limitation in the data processing system to occur, committing a portion of the stored data.

24. The computer-readable medium of claim 23 further having instructions for causing the portion of the stored data committed to include stored data through and including the end of the stored data for the last transaction within the computer code to be fully executed.

25. The computer-readable medium of claim 21 further having instructions for causing a processor to perform a method for processing a plurality of transactions within computer code being executed by a data processing system, the method comprising:

d) maintaining a pointer to the logical beginning of the stored data and maintaining a pointer to the end of the stored data for the last transaction within the computer code to be fully executed.

26. The computer-readable medium of claim 21 further having instructions for causing a processor to perform a method for processing a plurality of transactions within computer code being executed by a data processing system, the method comprising:

d) utilizing new value logging.

27. The computer-readable medium of claim 21 further having instructions for causing a processor to perform a method for processing a plurality of transactions within computer code being executed by a data processing system, the method comprising:

d) utilizing old value logging.

9. Evidence appendix

Not applicable.

10. Related proceedings appendix

Not applicable.